10. VC1

Program Name: VC1.java   Input File: vc1.dat

In distributed computing, there is a concept known as a "vector clock". Assume you have a distributed system consisting of k processes, and each process runs a series of events, and communicates with the other processes by sending messages. These events can either be local events, a send event, or a receive event. The vector clock of an event e contains, for each process, the entire history of events that "happen before" e. Instead of a long list of events though, you can show that there is one last event on each process that happens before e, so you can represent the entire history by storing the event number of the last event of each process that happens before e in a vector, creating a "vector clock". With this, we can build a relation between vector clocks called "happens-before", or à. The syntax we will use is: given p processes, a vector clock VC for an event e is an array of integers of length p, where VC[i] is the last event number on process i that **"happened-before"** e. For example, for event e = event 3 on process 1 (0-indexed), with 3 total processes, the vector clock could be [2, 3, 1], or [0, 3, 0], or [4, 1, 7], etc….
Vector clocks have the following properties with respect to the "happens-before" relation:
If two events are on the same process, then the earlier event à the later event.
If event a is the send event for a message M and event b is the receive event for a message M, then a à b.
If a à b and b à c, then a à c (transitivity).
Given two events a and b, if neither a à b nor b à a, then they are said to be "concurrent", or a || b.
.
Since vector clocks contain the last event that "happens-before" e on each process, given two vector clocks A and B for events a and b respectively, a à b when for all processes i, A[i] ≤ B[i]. In other words, for every entry i in the vector clock, the last event number A has for process i must be less than or equal to the last event number B has for process i.

Also, if no event on process i has à e, we say E[i] = 0.
Given the vector clocks of two events a and b, your job is to figure out whether a à b, b à a, or a || b (they are 'concurent' -- neither a à b nor b à a) .

Input
The first line will contain the number of test cases T.
Each test case contains three lines. The first line is the number of processes P. The second line is the vector clock A represented by P space-separated integers. The third line is the vector clock B, in the same format as A.

Output
Output "`A -> B`" if A à B, "`B -> A`" if B à A, and "`A || B`" if they are concurrent. It is guaranteed that A and B will never be equal, so exactly one of these is always true.

Constraints
```
1 <= T <= 10
0 <= P <= 20
1 <= each event number <= 10^9
```

Example Input File

```
3
2
1 1
2 1
3
3 3 3
1 2 3
2
1 2
2 1
```

Example Output to Screen

```
A -> B
B -> A
A || B
```

Explanation of the example

In the first test case, all numbers in A are ≤ all numbers in B. In the second case, all numbers in B are ≤ all numbers in A. In the third case, A[0] < B[0], but A[1] > B[1], so they are concurrent.